TEKNIK CUBE MAPPING DENGAN OPENGL

Eri Prasetyo[1] , Dian Kusuma Ningtyas[2], Prasetiyo[3]

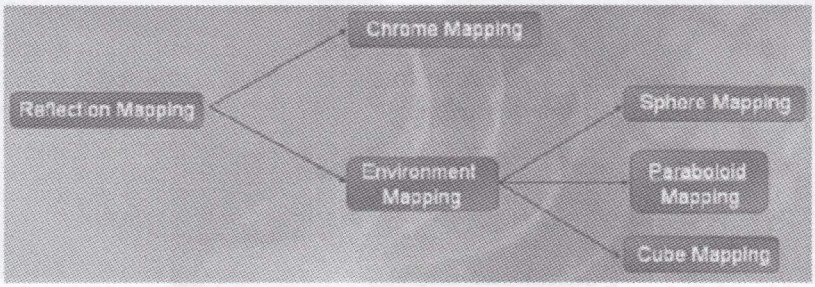
Abstrak:

Di bidang komputer grafik, Environment Mapping merupakan teknik untuk mensimulasikan sebuah obyek agar dapat merefleksikan lingkungan sekitarnya. Teknik ini pertama kali diajukan oleh Blinn dan Newell pada tahun 1976. Cube Mapping sebagai bagian dari metode Environment Mapping merepresentasikan lingkungan sekitarnya dengan cara "menempelkan" enam buah gambar yang berbeda di keenam sisi obyek. Hal ini membuat obyek seolah memiliki enam sisi pantul, yaitu depan, belakang, kanan, kiri, atas, dan bawah.

OpenGL sebagai kumpulan library, fungsi, dan prosedur untuk bidang komputer grafik telah mendukung Cube Mapping sebagai salah satu teknik Texture Mapping. Kemampuan OpenGL dalam mendukung Cube Mapping membuat dunia komputer grafik memiliki fitur tambahan untuk dapat lebih menghasilkan sesuatu yang lebih realistis. Keunggulan OpenGL yang platform-independent memungkinkan kita untuk membuat grafik yang dapat dijalankan di semua sistem operasi dengan hanya sedikit penyesuaian.

Kata Kunci : *komputer grafik, proyeksi, refleksi lingkungan, Texture Mapping.*

1. Pendahuluan - Reflection Mapping



Gambar. 1. Bagan Pembagian Metode Reflection Mapping

Reflection Mapping adalah teknik yang dapat membuat gambar/obyek menjadi terlihat semakin nyata dengan cara merefleksikan lingkungan sekitar di permukaan obyek. Dua metode Reflection Mapping yang dikenal adalah Chrome Mapping dan Environment Mapping.

Pada metode Chrome Mapping, refleksi/pantulan lingkungan sekitar obyek direpresentasikan dengan gambar yang dikaburkan (*blurred*) seperti halnya melihat pantulan pada benda-benda logam. Metode ini memberikan kesan mengkilap pada obyek.

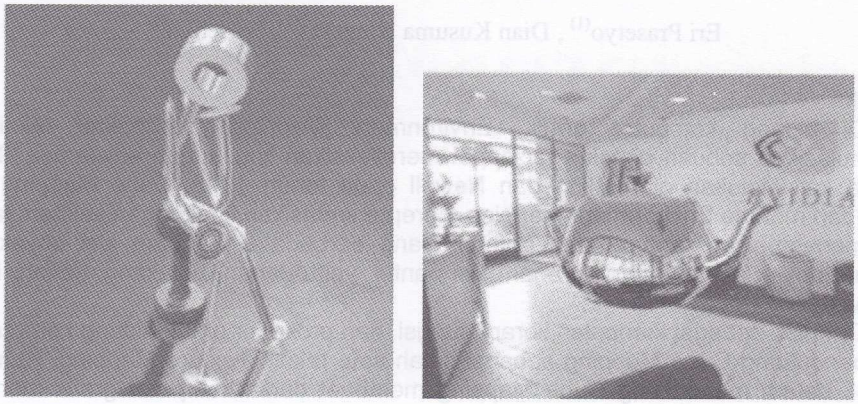
Metode lainnya, yaitu metode Environment Mapping merepresentasikan lingkungan sekitarnya dengan benar-benar "mencerminkan" lingkungannya. Tidak seperti metode Chrome Mapping yang hanya membuat obyek sekedar mengkilap, Environment Mapping memberikan kesan seolah-olah obyek tersebut merupakan "cermin" dari lingkungan sekitarnya.

2. Environment Mapping

Di bidang komputer grafik, Environment Mapping merupakan teknik untuk mensimulasikan sebuah obyek agar dapat merefleksikan lingkungan sekitarnya. Teknik ini pertama kali diajukan oleh Blinn dan Newell pada tahun 1976. Pada bentuk yang paling sederhana, teknik ini biasanya memakai obyek yang permukaannya terlihat seperti krom. Konsep dari teknik ini ialah menggunakan beberapa gambar yang diambil dari lingkungan sekitarnya ataupun gambar rekaan untuk dijadikan lingkungan yang akan direfleksikan oleh obyek.

---

[1] *Eri Prasetyo, Mahasiswa Program Sarjana Magister Universitas Gunadarma. Email : eri@staff.gunadarma.ac.id*
[2] *Dian Kusuma N ingtyas, Mahasiswa Program Sarjana Magister Universitas Gunadarma. Email : blue_dtctive@yahoo.com*
[3] *Prasetiyo, Mahasiswa Program Sarjana Magister Universitas Gunadarma. Email : prasetiyo@student.gunadarma.ac.id*

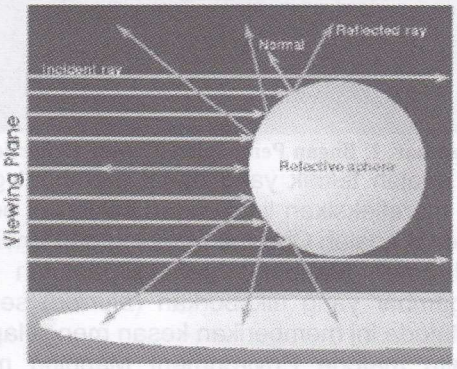


Gambar. 2. Chrome Mapping dan Environment Mapping

Ada beberapa teknik Environment Mapping, antara lain Sphere Mapping, Dual Paraboloid Mapping, dan Cube Mapping. Adapun yang akan dijelaskan lebih lanjut ialah teknik Cube Mapping.

2.1. Sphere Mapping

Sphere Mapping merupakan salah satu tipe dari Environment Mapping, di mana *irradiance image*[1] ekuivalen dengan apa yang mungkin terlihat pada *sphere* (bola) saat dilihat dengan proyeksi ortografik[2]. Konsep tersebut diilustrasikan pada gambar di bawah ini.



Gambar. 3. Konsep Sphere Mapping

Walaupun Sphere Mapping terasa meyakinkan, teknik ini belum sempurna benar. Idealnya, jika obyek yang akan direfleksikan berada dekat dengan obyek yang akan merefleksikan, refleksi yang didapat akan terlihat berbeda ketika dilihat dari titik yang berbeda pula. Tetapi, hal itu tidak akan terjadi jika menggunakan Sphere Mapping. Hasil dari Sphere Mapping hanya akan benar jika semua obyek yang akan direfleksikan berada jauh dari obyek yang merefleksikan (lihat gambar 4). Sehingga teknik ini membutuhkan gambar yang berbeda untuk setiap sudut pandang yang berbeda.

Sebagai akibat dari tidak tertutupnya semua permukaan obyek dengan gambar tekstur, teknik ini juga kadang menimbulkan "lubang" pada pinggiran obyek. Berikut gambar hasil Sphere Mapping dimana terlihat adanya "lubang" yang terbentuk.

2.2. Dual Paraboloid Mapping

Dual Paraboloid Mapping dapat mengatasi keterbatasan yang ada pada Sphere Mapping, tetapi teknik ini lebih rumit sebab membutuhkan 2 unit tekstur atau 2 tahap rendering.

---

[1] *Lingkungan sekitar yang berpotensi untuk dipantulkan oleh objek*

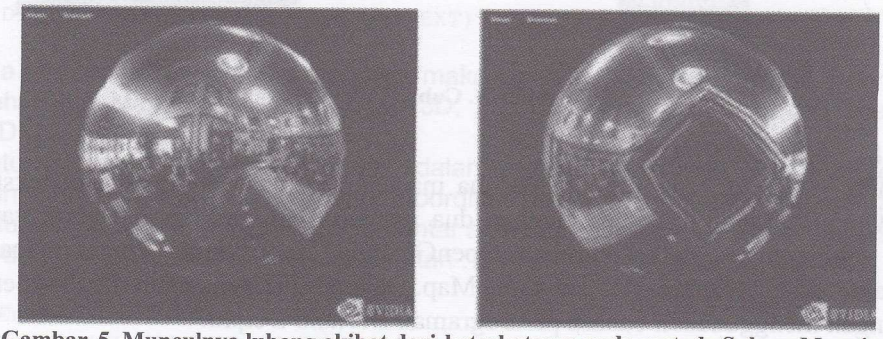
[2] *Proyeksi ortografik adalah cara untuk memproyeksikan obyek 3D ke dalam media 2D*

Teknik ini pertama kali diajukan dan dikembangkan oleh Wolfgang Heidrich dan Hans-Peter Seidel.

Keuntungan dari Dual Paraboloid Mapping, antara lain :

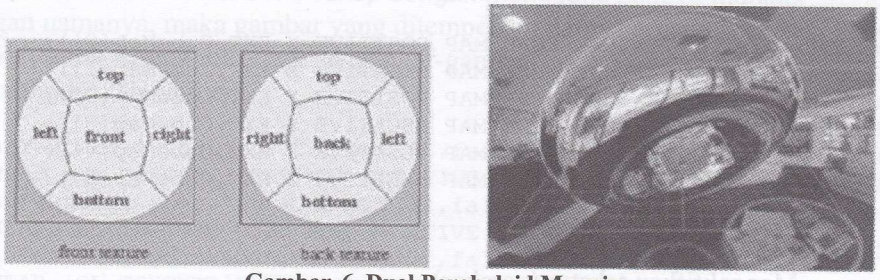
1) Dapat meng-*capture* lingkungan secara utuh.
2) Berbasis linear.

Gambar. 4. Kesalahan yang terjadi pada Sphere Mapping. Pada gambar sebelah kana n, tidak mungkin bisa melihat kolam pada posisi frontal sementara pada kenyataannya kolam berada di bawah obyek.



Gambar. 5. Munculnya lubang akibat dari keterbatasan pada metode Sphere Mapping

3) Cocok untuk *hardware* yang memiliki *dual-texture*, contohnya RIVA TNT.
4) *View independent.*



Gambar. 6. Dual Paraboloid Mapping

2.3. Cube Mapping

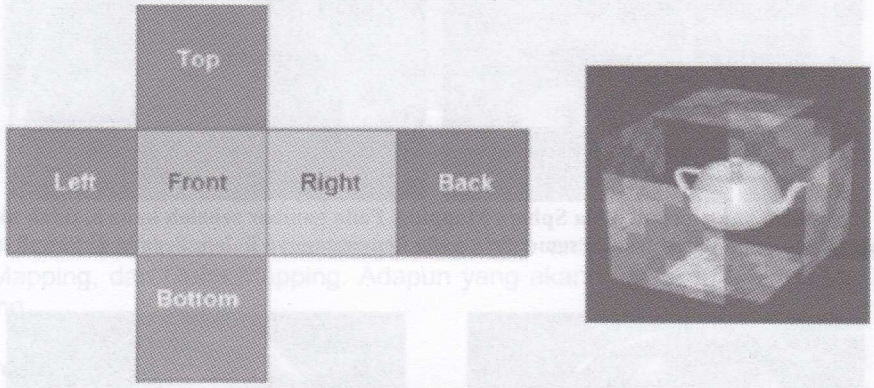
Cube Mapping sebagai bagian dari metode Environment Mapping merepresentasikan lingkungan sekitarnya dengan cara "menempelkan" enam buah gambar yang berbeda di keenam sisi obyek. Hal ini membuat obyek seolah memiliki enam sisi pantul[3], yaitu depan, belakang, kanan, kiri, atas, dan bawah (lihat gambar 7).

Cube Mapping muncul sebagai pengganti dua metode mapping sebelumnya. Hal-hal yang menjadi kelemahan dua metode terdahulu seperti ketergantungan sudut pandang (*view dependency*), keterbatasan cangkupan tekstur (*warping & distortion*), dan kerumitan penerapan

[3] *Sisi pantul adalah sisi obyek yang dapat memantulkan lingkungan sekitarnya*

menjadi alasan beralihnya teknik mapping ke Cube Mapping. Dengan mentransformasikan tekstur ke dalam enam sisi kubus, Cube Mapping lebih menawarkan kemudahan implementasi karena pantulan pada permukaan obyek cukup dikonsentrasikan di keenam sisi obyek.

Tidak seperti Dual Paraboloid Mapping, teknik Cube Mapping hanya membutuhkan satu unit tekstur[4] dan satu tahap rendering. Selain itu, teknik Cube Mapping tidak mengurangi resolusi gambar (teknik Sphere Mapping dan Dual Paraboloid Mapping dapat mengurangi resolusi gambar sampai 78% dari resolusi semula). Secara konsep, Cube Mapping memang lebih "*to the point*" dibandingkan dengan dua teknik lainnya. Namun, proses texturing pada Cube Mapping membutuhkan kemampuan yang lebih agar dapat mengakses enam gambar secara bersamaan.



Gambar. 7. Cube Mapping

3. Cube Mapping dalam OpenGL

Dahulu, OpenGL hanya mengenal dua macam Texture Mapping, yaitu tekstur 1D dan 2D. Mulai OpenGL versi 1.2 diperkenalkan dua macam Texture Mapping baru, yaitu 3D dan Cube Map sehingga secara keseluruhan OpenGL telah mendukung empat macam Texture Mapping, yaitu tekstur 1D, 2D, 3D, dan Cube Map. Dalam tulisan ini, akan dibahas pemrograman OpenGL dengan menggunakan bahasa pemrograman Borland Delphi.

Seperti yang telah dibahas sebelumnya, ide pokok Cube Mapping adalah memproyeksikan lingkungan sekitarnya dengan cara menempelkan gambar 2D di keenam sisi kubus. Untuk itu, OpenGL menggunakan perintah-perintah :

```
glBindTexture(GL TEXTURE CUBE MAP POSITIVE X EXT,CubeMap[0]);
glBindTexture(GL TEXTURE CUBE MAP NEGATIVE X EXT,CubeMap[1]);
glBindTexture(GL TEXTURE CUBE MAP POSITIVE Y EXT,CubeMap[2]);
glBindTexture(GL TEXTURE CUBE MAP NEGATIVE Y EXT,CubeMap[3]);
glBindTexture(GL TEXTURE CUBE MAP POSITIVE Z EXT,CubeMap[4]);
glBindTexture(GL TEXTURE CUBE MAP NEGATIVE Z EXT,CubeMap[5]);
```

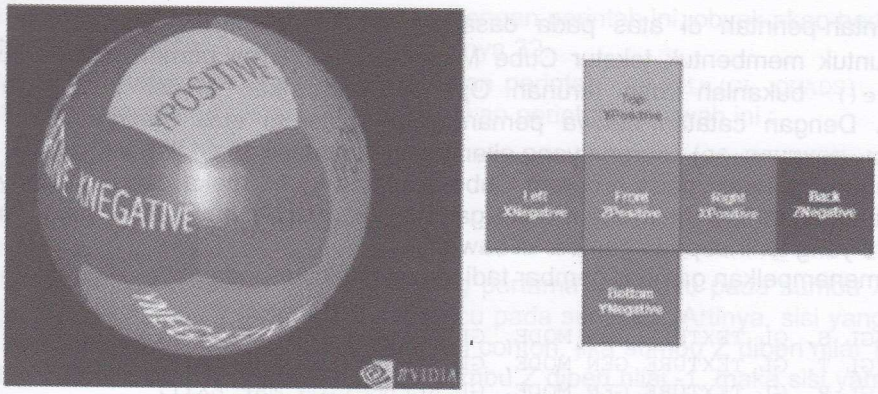
Penjelasan :

`glBindTexture` : Mengaktifkan tekstur (gambar).

`GL TEXTURE CUBE MAP POSITIVE{NEGATIVE} X{Y,Z} EXT` : Menempatkan gambar pada sumbu X/Y/Z positif/negatif. Akhiran EXT menandakan bahwa perintah/fungsi ini lintas platform (*platform independent*).

`CubeMap[0..5]` : Array tekstur / gambar yang akan ditempelkan (sebanyak enam buah).

[4] *Terdiri dari enam sisi/enam gambar*



Gambar. 8. Posisi Koordinat pada Cube Mapping

Untuk mengaktifkan fitur Cube Mapping, maka digunakan perintah :

```
glEnable(GL TEXTURE CUBE MAP EXT);
```

sebaliknya, untuk me-nonaktifkan fitur ini digunakan perintah :

```
glDisable (GL TEXTURE CUBE MAP EXT);
```

Bila terdapat lebih dari satu mapping, maka OpenGL memiliki prioritas[5]. Prioritas tertinggi dimiliki oleh Cube Mapping, diikuti dengan 3D, 2D, dan yang paling rendah adalah Texture Mapping 1D.

Satu hal pokok yang harus dicatat adalah terdapat perbedaan antara koordinat tekstur dan koordinat obyek pada CubeMapping. Koordinat obyek menggunakan koordinat Cartesian umum, yaitu +X, -X, +Y, -Y, +Z, dan -Z. Sementar a tekstur menggunakan koordinat STR (+S, -S, +T, -T, +R, dan -R). Untuk itu, digunakan perintah :

```
glEnable(GL TEXTURE GEN S);
glEnable(GL TEXTURE GEN T);
glEnable(GL TEXTURE GEN R);
```

Untuk menonaktifkan koordinat STR, cukup dengan mengganti `Enable` menjadi `Disable`.
Sesuai dengan namanya, maka gambar yang ditempelkan harus berbentuk persegi dengan dimensi (tinggi&lebar) sama. Untuk menetapkan gambar-gambar mana saja yang akan menempati sisi-sisi kubus, maka digunakan perintah :

```
LoadTexture('xpos.jpg',CubeMap[0],false,GL LINEAR MIPMAP LINEAR,GL
LINEAR, GL TEXTURE CUBE MAP POSITIVE X EXT);
LoadTexture('xneg.jpg',CubeMap[1],false,GL LINEAR MIPMAP LINEAR,GL
LINEAR, GL TEXTURE CUBE MAP NEGATIVE X EXT);
LoadTexture('ypos.jpg',CubeMap[2],false,GL LINEAR MIPMAP LINEAR,GL
LINEAR, GL TEXTURE CUBE MAP POSITIVE Y EXT);
LoadTexture('yneg.jpg',CubeMap[3],false,GL LINEAR MIPMAP LINEAR,GL
LINEAR, GL TEXTURE CUBE MAP NEGATIVE Y EXT);
LoadTexture('zpos.jpg',CubeMap[4],false,GL LINEAR MIPMAP LINEAR,GL
LINEAR, GL TEXTURE CUBE MAP POSITIVE Z EXT);
LoadTexture('zneg.jpg',CubeMap[5],false,GL LINEAR MIPMAP LINEAR,GL
LINEAR, GL TEXTURE CUBE MAP NEGATIVE Z EXT);
```

Dengan cara yang sama, untuk memuat gambar latar (background) digunakan perintah :

```
LoadTexture('back.jpg',Texture,false);
```

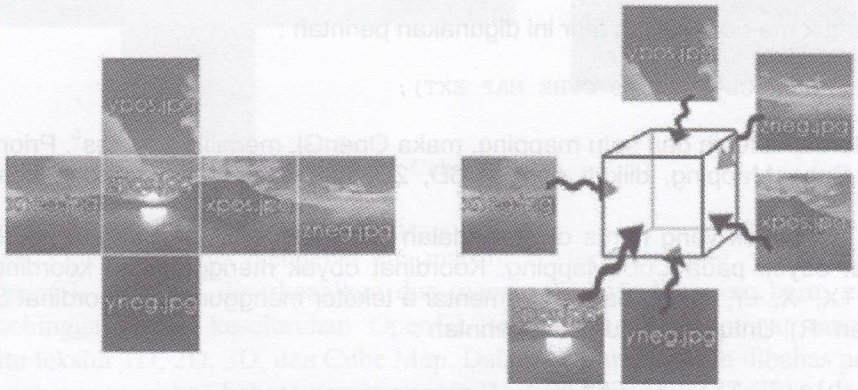
[5] *Untuk menentukan Texture Mapping mana yang akan dimunculkan terlebih dahulu.*

Perintah-perintah di atas pada dasarnya menempatkan file-file gambar yang akan digunakan untuk membentuk tekstur Cube Mapping pada array bernama `CubeMap[]`. Fungsi `LoadTexture()` bukanlah rutin turunan OpenGL, melainkan fungsi yang ada pada file Texture.pas. Dengan catatan bahwa pemanggilan fungsi di atas menggunakan perintah : `glEnable( GL TEXTURE 2D)` karena yang akan diimplementasikan adalah tekstur 2D.

Dengan mengacu pada program Cube Mapping kami, maka dengan adanya perintah-perintah di atas akan terbentuk enam buah gambar yang selanjutnya akan membentuk sebuah kubus seperti yang terlihat pada gambar di bawah ini:
Lalu, untuk menempelkan gambar-gambar tadi sebagai tekstur pada obyek digunakan perintah :

```
glTexGeni(GL S, GL TEXTURE GEN MODE, GL REFLECTION MAP EXT);
glTexGeni(GL T, GL TEXTURE GEN MODE, GL REFLECTION MAP EXT);
glTexGeni(GL R, GL TEXTURE GEN MODE, GL REFLECTION MAP EXT);
```

Selanjutnya, bagian empat akan menjelaskan jalannya program secara garis besar. Termasuk di dalamnya penjelasan lebih lanjut mengenai beberapa sintaks yang berhubungan dengan pembuatan tekstur Cube Mapping.



Gambar. 9. Cube Mapping yang terbentuk pada program

4. Alur Program

Program utama yang akan diuraikan pada bab ini adalah OpenGLApp.dpr yang ditulis dalam bahasa pemrograman Borland Delphi 7. Sebelum memulai program, dibutuhkan enam buah gambar sebagai map yang akan direfleksikan, dan satu buah gambar sebagai background atau "*fake environment*".

Awalnya, untuk inisialisasi OpenGL digunakan prosedur `glInit` yang berisi serangkaian perintah, seperti :

`glClearColor(0.0, 0.0, 0.0, 0.0)` : Untuk menentukan warna latar (*background*), untuk hal ini warna yang dihasilkan adalah hitam. Bentuk umum : `glClearColor(RedIndex, GreenIndex, BlueIndex, AlphaIndex).`

`glShadeModel(GL SMOOTH)` : Untuk menentukan modus bayangan (*shadow*), untuk hal ini yang dipilih adalah *smooth*.

`glEnable(GL DEPTH TEST)` : Untuk mengaktifkan memori buffer kedalaman (*Depth Buffer*).

Selain itu, perintah lain di luar prosedur `glInit` yang juga berhubungan dengan inisialisasi adalah :

`glClear(GL COLOR BUFFER BIT or GL DEPTH BUFFER BIT)`: Untuk membersihkan memori buffer warna atau memori buffer kedalaman dari keadaan sebelumnya.

`glLoadIdentity()`: Untuk mengembalikan posisi koordinat ke posisi normalnya.

Setelah tahap inisialisasi, maka langkah selanjutnya adalah membentuk obyek. Obyek yang akan kita bentuk adalah bola dan kubus yang berputar dengan bola sebagai poros putarnya. Perintah yang digunakan untuk menggambar bola ialah

`gluSphere(SphereQuadratic, 0.8, 32, 32)`. Dengan perintah ini, obyek akan berbentuk bola yang diameternya 0.8, bersisi 32, dan jumlah irisannya 32.

Untuk mulai menggambar kubus, digunakan perintah `glBegin(GL QUADS)`. Sedangkan untuk memulai menggambar sisi-sisi kubus, digunakan perintah di bawah ini :

```
glNormal3f( 0.0, 0.0, 1.0);
glVertex3f(-0.25, -1.0, 1.0);
glVertex3f( 0.25, -1.0, 1.0);
glVertex3f( 0.25, 1.0, 1.0);
glVertex3f(-0.25, 1.0, 1.0);
```

Pada bagian `glNormal3f`, angka 0.0 yang pertama mengacu pada sumbu X dan yang kedua mengacu pada sumbu Y. Angka 1.0 mengacu pada sumbu Z. Artinya, sisi yang digambar mempunyai poros normal sumbu Z positif. Sebagai contoh, jika sumbu Z diberi nilai 1, maka sisi yang digambar ialah sisi depan. Sebaliknya jika sumbu Z diberi nilai -1, maka sisi yang digambar ialah sisi belakang. Kemudian barulah ditentukan vertex-vertexnya yang juga mengacu pada sumbu X, Y, dan Z. Penggambaran sisi ini diulangi terus hingga bentuk kubus yang diinginkan tercapai, dan diakhiri dengan perintah `glEnd()`.

Proses pembentukan obyek di atas dapat direpresentasikan sebagai matrix yang ditampung dalam sebuah penampung (*stack*). Prinsip kerjanya adalah, OpenGL akan menganggap semua masukan pasangan vertex sebagai matrix dan akan ditampung ke dalam suatu penampung. Untuk itu, OpenGL menyediakan perintah `glPushMatrix` dan sebaliknya jika tiba saatnya untuk menampilkan obyek, maka perintah `glPopMatrix` dapat digunakan. Pada umumnya, perintah `glPushMatrix` dan `glPopMatrix` berpasangan dengan `SwapBuffers/glFlush` sebagai "eksekutor"[6] nya.

Untuk pemuatan gambar latar, yaitu `back.jpg`, digunakan perintah:

```
LoadTexture('back.jpg',Texture,false).
```

Sedangkan untuk memuat masing-masing gambar tekstur Cube Map, digunakan rangkaian perintah berikut :

`LoadTexture('xpos.jpg',CubeMap[0],false,GL LINEAR MIPMAP LINEAR,GL LINEAR, GL TEXTURE CUBE MAP POSITIVE X EXT)`: Untuk memuat gambar xpos.jpg pada array `CubeMap[0]`.

`LoadTexture('xneg.jpg',CubeMap[1],false,GL LINEAR MIPMAP LINEAR,GL LINEAR, GL TEXTURE CUBE MAP NEGATIVE X EXT)`: Untuk memuat gambar xneg.jpg pada array `CubeMap[1]`.

`LoadTexture('ypos.jpg',CubeMap[2],'false,GL LINEAR MIPMAP LINEAR,GL LINEAR, GL TEXTURE CUBE MAP POSITIVE Y EXT)`: Untuk memuat gambar ypos.jpg pada array `CubeMap[2]`.

`LoadTexture('yneg.jpg',CubeMap[3],false,GL LINEAR MIPMAP LINEAR,GL LINEAR, GL TEXTURE CUBE MAP NEGATIVE Y EXT)`: Untuk memuat gambar yneg.jpg pada array `CubeMap[3]`.

`LoadTexture('zpos.jpg',CubeMap[4],false,GL LINEAR MIPMAP LINEAR,GL LINEAR, GL TEXTURE CUBE MAP POSITIVE Z EXT)`: Untuk memuat gambar zpos.jpg pada array `CubeMap[4]`.

`LoadTexture('zneg.jpg',CubeMap[5],false,GL LINEAR MIPMAP LINEAR,GL LINEAR, GL TEXTURE CUBE MAP NEGATIVE Z EXT)`: Untuk memuat gambar zneg.jpg pada array `CubeMap[5]`.

Selanjutnya, pengaturan posisi set obyek dilakukan dengan menggunakan perintah :

glTranslatef(0.0,0.0,-5)

Makin besar nilai sumbu Z, makin besar gambar yang terlihat. Namun jika sumbu Z bernilai positif, maka set obyek akan nampak kosong sebab gambar berada di luar jangkauan layar. Sebaliknya, semakin kecil nilainya, maka set obyek juga akan terlihat mengecil (menjauh).

Selanjutnya, agar kubus dapat berotasi, digunakan perintah : `glRotatef(yAngle, 0.0, 1.0, 0.0)`. Sumbu Y diberi nilai 1, artinya kubus akan berotasi sebesar yAngle derajat terhadap sumbu Y (horizontal).

---

[6] *SwapBuffers/glFlush adalah perintah yang digunakan untuk menampilkan gambar di layar.*

Kecepatan rotasi kubus diatur dengan rumus `yAngle := yAngle + ySpeed` sehingga kecepatannya dapat berubah-ubah sesuai nilai dari variabel `ySpeed`. Dengan menjalankan perintah: `if(keys[VK RIGHT]) then ySpeed := ySpeed + 0.002`, jika tombol kanan ditekan maka kecepatan rotasi akan bertambah sebesar 0,002 fps. Sedangkan dengan mengganti `(keys[VK RIGHT])` menjadi `(keys[VK LEFT])` dan `then ySpeed := ySpeed + 0.002` menjadi `then ySpeed := ySpeed - 0.002`, maka jika tombol kiri ditekan kecepatan rotasi akan berkurang sebesar 0,002 fps dan dapat berubah haluan (berputar ke arah kiri).

Jika user menggerakkan mouse, maka pergerakan tersebut dianggap tidak ada (tidak terjadi apa-apa). Begitu juga dengan penekanan mouse, baik itu tombol kiri, tengah, maupun kanan, akan dianggap tidak ada dengan perintah yang digunakan:

```
case MouseButton of
1: MouseButton := 0;
2: MouseButton := 0;
3: MouseButton := 0;
end;
```

Jika user mengubah ukuran window, maka ukuran gambar juga akan disesuaikan. Jika ukuran window diperbesar maka gambar juga ikut membesar, begitu juga sebaliknya. Perintah yang digunakan :

```
if (Height = 0) then
Height := 1;
glViewport(0, 0, Width, Height);
glMatrixMode(GL PROJECTION);
glLoadIdentity();
gluPerspective(45.0, Width/Height, 1.0, 100.0);
glMatrixMode(GL MODELVIEW);
glLoadIdentity();
```

Perintah-perintah tersebut akan terus dijalankan sampai tombol Escape ditekan. Untuk itu digunakan perintah: `if (keys[VK ESCAPE]) then finished := True.`

5. Simpulan dan Penutup

Bila dibandingkan dengan Sphere Mapping dan Dual Paraboloid Mapping, secara konsep Cube Mapping lebih to the point dan praktis. Hal ini dapat dilihat dari kebutuhan Cube Mapping yang hanya memerlukan satu unit tekstur dan satu tahap rendering. Selain itu, teknik Cube Mapping tidak mengurangi resolusi gambar (teknik Sphere Mapping dan Dual Paraboloid Mapping dapat mengurangi resolusi gambar sampai 78% dari resolusi semula). Intinya, hal-hal yang menjadi kelemahan Sphere Mapping dan Dual Paraboloid Mapping seperti ketergantungan sudut pandang (view dependency), keterbatasan cangkupan tekstur (warping and distortion), dan kerumitan penerapan menjadi alasan beralihnya teknik mapping ke Cube Mapping. Namun, dibutuhkan kemampuan lebih pada proses texturing Cube Mapping untuk dapat mengakses enam gambar sekaligus. OpenGL sebagai kumpulan rutin dan fungsi yang banyak dipakai di bidang komputer grafik telah menyediakan fasilitas Cube Mapping. Dengan menggunakan rutin tersebut, para pemrogram komputer grafik dapat membuat grafis yang lebih realistis dari sekedar gambar 2D dan 3D biasa. Portabilitas OpenGL sebagai kumpulan library memungkinkan untuk diakses melalui bahasa pemrograman (dengan menyertakan *library* OpenGL di direktori *library* bahasa tersebut) dan dijalankan pada platform komputer apapun.

6. Daftar Pustaka

[1]http://en.wikipedia.org/wiki/Reflection mapping, 2007.
[2]http://www.opengl.org/resources/code/samples/sig99/advanced99/notes/node174.html, 2007.
[3]http://www.opengl.org/resources/code/samples/sig99/advanced99/notes/node176.html, 2007.
[4]http://www.opengl.org/resources/code/samples/sig99/advanced99/notes/node183.html, 2007.
[5]http://developer.nvidia.com/object/cube map ogl tutorial.html, 2007.
[6]http://www.developer.com/lang/other/article.php/2169281, 2007.
[7]http://developer.nvidia.com/object/dual paraboloid env mapping.html, 2007.
[8]http://www.glprogramming.com/red/, 2007.

[9]http://developer.nvidia.com/object/IO 20010903 4702.html, 2007.
[10]http://www.nvidia.com/object/feature cube.html, 2007.
[11]F.S. Hill. *Computer Graphics Using OpenGL*. Prentice Hall Inc, 2001.